10 |调优一筹莫展，配置项速查手册让你事半功倍！（下）

2021-04-05 吴磊

Spark性能调优实战 进入课程 >

讲述：吴磊
时长 20:00 大小 18.32M

你好，我是吴磊。

上一讲，我们讲了硬件资源类的配置项。这一讲，我们继续说说 Shuffle 类和 Spark SQL 大类都有哪些配置项，它们的含义和作用，以及它们能解决的问题。同时，和上一讲一样，我们今天讲到的配置项也全部会围绕 Executors 展开。

Shuffle 类配置项

首先，我们来说说 Shuffle 类。纵观 Spark 官网的 Configuration 页面，你会发现[节 Shuffle 执行性能的配置项真是寥寥无几。其实这也很好理解，因为一旦 Shuffle 成为应用中不可或缺的一环，想要优化 Shuffle 本身的性能，我们能做的微乎其微。

不过，我们也不是完全束手无策。我们知道，Shuffle 的计算过程分为 Map 和 Reduce 这两个阶段。其中，Map 阶段执行映射逻辑，并按照 Reducer 的分区规则，将中间数据写入到本地磁盘；Reduce 阶段从各个节点下载数据分片，并根据需要实现聚合计算。

那么，我们就可以通过 spark.shuffle.file.buffer 和 spark.reducer.maxSizeInFlight 这两个配置项，来分别调节 Map 阶段和 Reduce 阶段读写缓冲区的大小。具体该怎么做呢？我们一一来看。

配置项	含义	计算端
spark.shuffle.file.buffer	Map输出端的写入缓冲区大小	Map端
spark.reducer.maxSizeInFlight	Reduce输入端的读取缓冲区大小	Reduce端

缓冲区相关配置项

首先，在 Map 阶段，计算结果会以中间文件的形式被写入到磁盘文件系统。同时，为了避免频繁的 I/O 操作，Spark 会把中间文件存储到写缓冲区（Write Buffer）。**这个时候，我们可以通过设置 spark.shuffle.file.buffer 来扩大写缓冲区的大小，缓冲区越大，能够缓存的落盘数据越多，Spark 需要刷盘的次数就越少，I/O 效率也就能得到整体的提升。**

其次，在 Reduce 阶段，因为 Spark 会通过网络从不同节点的磁盘中拉取中间文件，它们又会以数据块的形式暂存到计算节点的读缓冲区（Read Buffer）。缓冲区越大，可以暂存的数据块越多，在数据总量不变的情况下，拉取数据所需的网络请求次数越少，单次请求的网络吞吐越高，网络 I/O 的效率也就越高。**这个时候，我们就可以通过 spark.reducer.maxSizeInFlight 配置项控制 Reduce 端缓冲区大小，来调节 Shuffle 过程中的网络负载。**

事实上，对 Shuffle 计算过程的优化牵扯到了全部的硬件资源，包括 CPU、内存、磁盘和网络。因此，我们上一讲汇总的关于 CPU、内存和硬盘的配置项，也同样可以作用在 Map 和 Reduce 阶段的内存计算过程上。

除此之外，Spark 还提供了一个叫做 **spark.shuffle.sort.bypassMergeThreshold** 的配置项，去处理一种特殊的 Shuffle 场景。

配置项	含义
spark.shuffle.sort.bypassMergeThreshold	Map阶段不进行排序的分区阈值

极客时间

Reduce端相关配置项

自 1.6 版本之后，Spark 统一采用 Sort shuffle manager 来管理 Shuffle 操作，在 Sort shuffle manager 的管理机制下，无论计算结果本身是否需要排序，Shuffle 计算过程在 Map 阶段和 Reduce 阶段都会引入排序操作。

这样的实现机制对于 repartition、groupBy 这些操作就不太公平了，这两个算子一个是对原始数据集重新划分分区，另一个是对数据集进行分组，压根儿就没有排序的需求。所以，Sort shuffle manager 实现机制引入的排序步骤反而变成了一种额外的计算开销。

因此，**在不需要聚合，也不需要排序的计算场景中，我们就可以通过设置 spark.shuffle.sort.bypassMergeThreshold 的参数，来改变 Reduce 端的并行度**（默认值是 200）。当 Reduce 端的分区数小于这个设置值的时候，我们就能避免 Shuffle 在计算过程引入排序。

Spark SQL 大类配置项

接下来，我们再来说说 Spark SQL 的相关配置项。在官网的Configuration 页面中，Spark SQL 下面的配置项还是蛮多的，其中对执行性能贡献最大的，当属 AQE（Adaptive query execution，自适应查询引擎）引入的那 3 个特性了，也就是自动分区合并、自动数据倾斜处理和 Join 策略调整。因此，关于 Spark SQL 的配置项，咱们围绕着这 3 个特性去汇总。

首先我们要知道，**AQE 功能默认是禁用的，想要使用这些特性，我们需要先通过配置项 spark.sql.adaptive.enabled 来开启 AQE**，具体的操作如下：

配置项	含义
spark.sql.adaptive.enabled	是否启用AQE（Adaptive query execution），默认禁用

极客时间

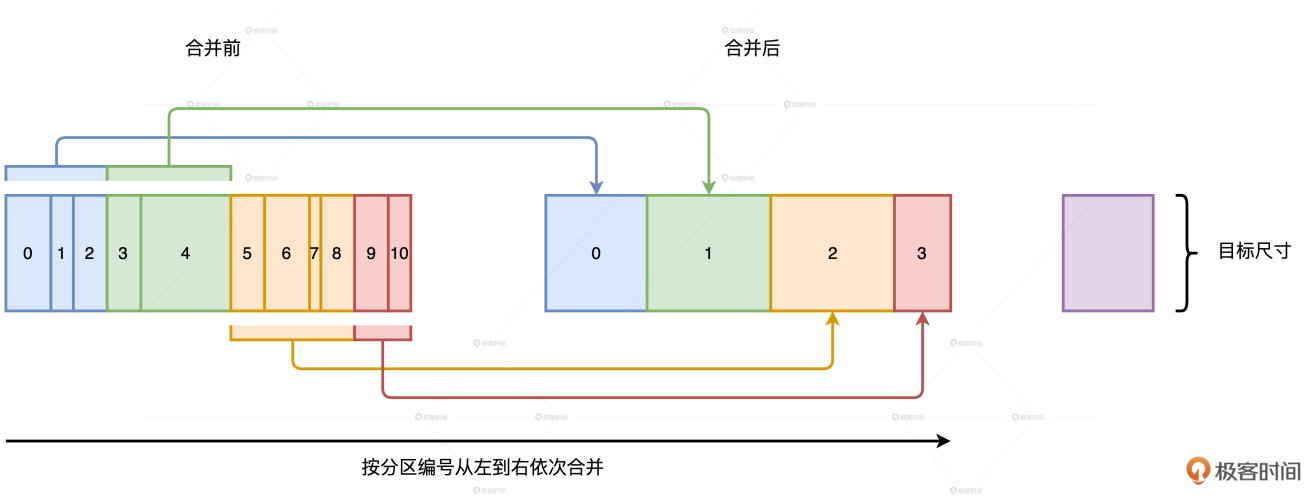
启用AQE的配置项

因为这 3 个特性的原理我们在开发原则那一讲说过，这里我会先带你简单回顾一下，然后我们重点来讲，这些环节对应的配置项有哪些。

哪些配置项与自动分区合并有关?

分区合并的场景用一句概括就是，在 Shuffle 过程中，因为数据分布不均衡，导致 Reduce 阶段存在大量的小分区，这些小分区的数据量非常小，调度成本很高。

那么问题来了，AQE 是如何判断某个分区是不是足够小，到底需不需要合并的呢？另外，既然是对多个分区进行合并，自然就存在一个收敛条件的问题，如果一直不停地合并下去，整个分布式数据集最终就会合并为一个超级大的分区。简单来说，就是：“分区合并从哪里开始，又到哪里结束呢？”



分区合并示意图

我们一起来看一下 AQE 分区合并的工作原理。如上图所示，对于所有的数据分区，无论大小，AQE 按照分区编号从左到右进行扫描，边扫描边记录分区尺寸，当相邻分区的尺寸之和大于“目标尺寸”时，AQE 就把这些扫描过的分区进行合并。然后，继续向右扫描，并采用同样的算法，按照目标尺寸合并剩余分区，直到所有分区都处理完毕。

总的来说就是，**AQE 事先并不判断哪些分区足够小，而是按照分区编号进行扫描，当扫描量超过“目标尺寸”时，就合并一次。**我们发现，这个过程中的关键就是“目标尺寸”的确定，它的大小决定了合并之后分布式数据集的分散程度。

那么，“目标尺寸”由什么来决定的呢？Spark 提供了两个配置项来共同决定分区合并的“目标尺寸”，它们分别是 spark.sql.adaptive.advisoryPartitionSizeInBytes 和 spark.sql.adaptive.coalescePartitions.minPartitionNum。

配置项	含义
spark.sql.adaptive.coalescePartitions.enabled	是否启用AQE中的自动分区合并，默认启用
spark.sql.adaptive.advisoryPartitionSizeInBytes	由开发者指定分区合并后的推荐尺寸
spark.sql.adaptive.coalescePartitions.minPartitionNum	分区合并后，数据集的分区数不能低于该值

极客时间

AQE自动分区合并相关配置项

其中，第一个参数 advisoryPartitionSizeInBytes 是开发者建议的目标尺寸，第二个参数 minPartitionNum 的含义是合并之后的最小分区数，假设它是 200，就说明合并之后的分区数量不能小于 200。这个参数的目的就是避免并行度过低导致 CPU 资源利用不充分。

结合 Shuffle 后的数据集尺寸和最小分区数限制，我们可以反推出来每个分区的平均大小，咱们暂且把它记为 #partitionSize。分区合并的目标尺寸取 advisoryPartitionSizeInBytes 与 #partitionSize 之间的最小值。

这么说比较抽象，我们来举个例子。假设，Shuffle 过后数据大小为 20GB，minPartitionNum 设置为 200，反推过来，每个分区的尺寸就是 20GB / 200 = 100MB。再假设，advisoryPartitionSizeInBytes 设置为 200MB，最终的目标分区尺寸就是取（100MB，200MB）之间的最小值，也就是 100MB。因此你看，并不是你指定了 advisoryPartitionSizeInBytes 是多少，Spark 就会完全尊重你的意见，我们还要考虑 minPartitionNum 的设置。

哪些配置项与自动数据倾斜处理有关？

再来说说数据倾斜，在数据关联（Data Joins）的场景中，当 AQE 检测到倾斜的数据分区时，会自动进行拆分操作，把大分区拆成多个小分区，从而避免单个任务的数据处理量过大。不过，Spark 3.0 版本发布的 AQE，暂时只能在 Sort Merge Join 中自动处理数据倾斜，其他的 Join 实现方式如 Shuffle Join 还有待支持。

那么，AQE 如何判定数据分区是否倾斜呢？它又是怎么把大分区拆分成多个小分区的？

配置项	含义
spark.sql.adaptive.skewJoin.enabled	是否开启AQE中自动处理数据倾斜功能，默认开启
spark.sql.adaptive.skewJoin.skewedPartitionFactor	判定数据分区是否倾斜的比例系数
spark.sql.adaptive.skewJoin.skewedPartitionThresholdInBytes	判定数据分区是否倾斜的最低阈值
spark.sql.adaptive.advisoryPartitionSizeInBytes	以字节为单位，拆分倾斜分区的数据粒度

极客时间

AQE数据倾斜处理相关配置项

首先，**分区尺寸必须要大于 spark.sql.adaptive.skewJoin.skewedPartitionThresholdInBytes 参数的设定值，才有可能被判定为倾斜分区。然后，AQE 统计所有数据分区大小并排序，取中位数作为放大基数，尺寸大于中位数一定倍数的分区会被判定为倾斜分区，中位数的放大倍数也是由参数 spark.sql.adaptive.skewJoin.skewedPartitionFactor 控制。**

接下来，我们还是通过一个例子来理解。假设数据表 A 有 3 个分区，分区大小分别是 80MB、100MB 和 512MB。显然，这些分区按大小个排序后的中位数是 100MB，因为 skewedPartitionFactor 的默认值是 5 倍，所以大于 100MB * 5 = 500MB 的分区才有可能被判定为倾斜分区。在我们的例子中，只有最后一个尺寸是 512MB 的分区符合这个条件。

这个时候，Spark 还不能完全判定它就是倾斜分区，还要看 skewedPartitionThresholdInBytes 配置项，这个参数的默认值是 256MB。对于那些满足中位数条件的分区，必须要大于 256MB，Saprk 才会把这个分区最终判定为倾斜分区。假设 skewedPartitionThresholdInBytes 设定为 1GB，那在我们的例子中，512MB 那个

大分区，Spark 也不会把它看成是倾斜分区，自然也就不能享受到 AQE 对于数据倾斜的优化处理。

检测到倾斜分区之后，接下来就是对它拆分，拆分的时候还会用到 advisoryPartitionSizeInBytes 参数。假设我们将这个参数的值设置为 256MB，那么，刚刚那个 512MB 的倾斜分区会以 256MB 为粒度拆分成多份，因此，这个大分区会被拆成 2 个小分区（ 512MB / 256MB =2）。拆分之后，原来的数据表就由 3 个分区变成了 4 个分区，每个分区的尺寸都不大于 256MB。

哪些配置项与 Join 策略调整有关?

最后，咱们再来说说数据关联（Joins）。数据关联可以说是数据分析领域中最常见的操作，Spark SQL 中的 Join 策略调整，它实际上指的是，把会引入 Shuffle 的 Join 方式，如 Hash Join、Sort Merge Join，“降级”（Demote）为 Broadcast Join。

Broadcast Join 的精髓在于“以小博大”，它以广播的方式将小表的全量数据分发到集群中所有的 Executors，大表的数据不需要以 Join keys 为基准去 Shuffle，就可以与小表数据原地进行关联操作。Broadcast Join 以小表的广播开销为杠杆，博取了因消除大表 Shuffle 而带来的巨大性能收益。可以说，Broadcast Join 把“杠杆原理”应用到了极致。

在 Spark 发布 AQE 之前，开发者可以利用 spark.sql.autoBroadcastJoinThreshold 配置项对数据关联操作进行主动降级。这个参数的默认值是 10MB，参与 Join 的两张表中只要有一张数据表的尺寸小于 10MB，二者的关联操作就可以降级为 Broadcast Join。为了充分利用 Broadcast Join“以小博大”的优势，你可以考虑把这个参数值调大一些，2GB 左右往往是个不错的选择。

配置项	含义
spark.sql.autoBroadcastJoinThreshold	数据表采用Broadcast Join实现方式的最低阈值

极客时间

AQE推出之前Join策略相关配置项

不过，autoBroadcastJoinThreshold 这个参数虽然好用，但是有两个让人头疼的短板。

一是可靠性较差。尽管开发者明确设置了广播阈值，而且小表数据量在阈值以内，但 Spark 对小表尺寸的误判时有发生，导致 Broadcast Join 降级失败。

二来，预先设置广播阈值是一种静态的优化机制，它没有办法在运行时动态对数据关联进行降级调整。一个典型的例子是，两张大表在逻辑优化阶段都不满足广播阈值，此时 Spark SQL 在物理计划阶段会选择 Shuffle Joins。但在运行时期间，其中一张表在 Filter 操作之后，有可能出现剩余的数据量足够小，小到刚好可以降级为 Broadcast Join。在这种情况下，静态优化机制就是无能为力的。

AQE 很好地解决了这两个头疼的问题。**首先，AQE 的 Join 策略调整是一种动态优化机制，对于刚才的两张大表，AQE 会在数据表完成过滤操作之后动态计算剩余数据量，当数据量满足广播条件时，AQE 会重新调整逻辑执行计划，在新的逻辑计划中把 Shuffle Joins 降级为 Broadcast Join。再者，运行时的数据量估算要比编译时准确得多，因此 AQE 的动态 Join 策略调整相比静态优化会更可靠、更稳定。**

配置项	含义
spark.sql.adaptive.nonEmptyPartitionRatioForBroadcastJoin	非空分区比例小于该值，才会调整Join策略

极客时间

AQE推出之后Join策略相关配置项

不过，启用动态 Join 策略调整还有个前提，也就是要满足 nonEmptyPartitionRatioForBroadcastJoin 参数的限制。这个参数的默认值是 0.2，大表过滤之后，非空的数据分区占比要小于 0.2，才能成功触发 Broadcast Join 降级。

这么说有点绕，我们来举个例子。假设，大表过滤之前有 100 个分区，Filter 操作之后，有 85 个分区内的数据因为不满足过滤条件，在过滤之后都变成了没有任何数据的空分区，另外的 15 个分区还保留着满足过滤条件的数据。这样一来，这张大表过滤之后的非空分区占比是 15 / 100 = 15%，因为 15% 小于 0.2，所以这个例子中的大表会成功触发 Broadcast Join 降级。

相反，如果大表过滤之后，非空分区占比大于 0.2，那么剩余数据量再小，AQE 也不会把 Shuffle Joins 降级为 Broadcast Join。因此，如果你想要充分利用 Broadcast Join 的优势，可以考虑把这个参数适当调高。

小结

今天这一讲，我们深入探讨了 Shuffle 类和 Spark SQL 大类两类配置项，以及每个配置项可以解决的问题。

对于 Shuffle 类我们要知道，在 Shuffle 过程中，对于不需要排序和聚合的操作，我们可以通过控制 spark.shuffle.sort.bypassMergeThreshold 参数，来避免 Shuffle 执行过程中引入的排序环节，从而避免没必要的计算开销。

对于 Spark SQL 大类我们首先要知道，AQE 默认是禁用状态，要充分利用 AQE 提供的 3 个特性，就是自动分区合并、数据倾斜处理和 Join 策略调整，我们需要把 spark.sql.adaptive.enabled 置为 true。

除此之外，AQE 的 3 个特性各自都有相对应的配置项，需要我们单独调整。

> AQE 中的自动分区合并过程与我们预想的不太一样。QE 事先并不判断哪些分区足够小，而是按照分区编号进行扫描，当扫描量超过“目标尺寸”时就合并一次。目标尺寸由 advisoryPartitionSizeInBytes 和 coalescePartitions.minPartitionNum 两个参数共同决定。
>
> AQE 能够自动处理 Sort Merge Join 场景中的数据倾斜问题。首先根据所有分区大小的中位数，以及放大倍数 skewedPartitionFactor 来检测倾斜分区，然后以 advisoryPartitionSizeInBytes 为粒度对倾斜分区进行拆分。
>
> AQE 动态 Join 策略调整可以在运行时将 Shuffle Joins 降级为 Broadcast Join，同时，运行时的数据量估算要比编译时准确得多，因此相比静态优化会更可靠。不过，需要我们注意的是，Shuffle 过后非空分区占比要小于 nonEmptyPartitionRatioForBroadcastJoin 才能触发 Broadcast Join 的降级优化。

好啦，经过这两讲的学习，我们一起汇总出了 Spark 中与性能调优息息相关的所有配置项，为了方便你快速查阅，我把它们合并在了一张文稿的表格中，希望你能在工作中好好利用起来。

《Spark 性能调优实战》配置项速查手册

配置项分类	子类细分	配置项	含义
硬件资源类	CPU	spark.cores.max	集群范围内满配CPU核数
		spark.executor.cores	单个Executor内CPU核数
		spark.task.cpus	单个任务消耗的CPU核数
		spark.default.parallelism	未指定分区数时的默认并行度
		spark.sql.shuffle.partitions	数据关联、聚合操作中Reducer的并行度
	内存	spark.executor.memory	单个Executor内堆内内存总大小
		spark.memory.offHeap.enabled	是否启用堆外内存，默认为False
		spark.memory.offHeap.size	单个Executor内堆外内存总大小
		spark.memory.fraction	堆内内存中，用于缓存RDD和执行计算的内存比例
		spark.memory.storageFraction	用于缓存RDD的内存占比，执行内存占比为 1 – spark.memory.storageFraction
		spark.rdd.compress	RDD缓存是否压缩，默认不压缩
	磁盘	spark.local.dir	用于缓存RDD和Shuffle中间文件的磁盘目录
Shuffle类	Map端	spark.shuffle.file.buffer	Map输出端的写入缓冲区大小
	Reduce端	spark.reducer.maxSizeInFlight	Reduce输入端的读取缓冲区大小
		spark.shuffle.sort.bypassMergeThreshold	Map阶段不进行排序的分区阈值
Spark SQL大类	AQE	spark.sql.adaptive.enabled	是否启用AQE（Adaptive query execution），默认禁用
	AQE分区合并	spark.sql.adaptive.coalescePartitions.enabled	是否启用AQE中的自动分区合并，默认启用
		spark.sql.adaptive.advisoryPartitionSizeInBytes	由开发者指定分区合并后的推荐尺寸
		spark.sql.adaptive.coalescePartitions.minPartitionNum	分区合并后，数据集的分区数不能低于该值
	AQE数据倾斜	spark.sql.adaptive.skewJoin.enabled	是否开启AQE中自动处理数据倾斜的功能，默认开启
		spark.sql.adaptive.skewJoin.skewedPartitionFactor	判定数据分区是否倾斜的比例系数
		spark.sql.adaptive.skewJoin.skewedPartitionThresholdInBytes	判定数据分区是否倾斜的最低阈值
		spark.sql.adaptive.advisoryPartitionSizeInBytes	以字节为单位，拆分倾斜分区的数据粒度
	AQE Join策略	spark.sql.autoBroadcastJoinThreshold	数据表采用Broadcast Join实现方式的最低阈值
		spark.sql.adaptive.nonEmptyPartitionRatioForBroadcastJoin	非空分区比例小于该值，才会调整Join策略

极客时间

每日一练

1. AQE 的分区合并算法略显简单粗暴，如果让你来重新实现分区合并特性的话，你都有哪些思路呢？

2. AQE 中数据倾斜的处理机制，你认为有哪些潜在的隐患？

期待在留言区看到你的思考和答案，也欢迎你把这份调优手册分享给你的朋友们，我们下一讲见！

提建议

更多课程推荐

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 09 | 调优一筹莫展，配置项速查手册让你事半功倍！（上）

下一篇 11 | Shuffle的工作原理：为什么说Shuffle是一时无两的性能杀手？

精选留言 (15)

写留言

来世愿做友人 A
2021-04-07

请教老师一个问题，对于小文件合并，上边说的是某个 executor 的大小排序后合并。比如两个 executorA 和 B，分别有两个 task 运行，groupby 并且各自产生了3个分区，分别是 a.0,a.1,a.2 和 b.0,b.1,b.2，在没有合并小分区的情况下，reduce端会有三个任务拉取各自的012分区。但是，打开小分区合并，在满足合并的条件下，a.0和a.1合并成 a.01，b.1和b.2合并成b.12。这时候两个 task 各有两个分区，但是他们的分区 key 相当于...
展开

作者回复: 好问题，思考的很深入，你是第一个问这个问题的同学，先赞一个～

首先，分区合并是在reduce阶段（这块原文我可能没说清楚，我的锅），也就是每个reduce task把数据拉回来，aqe发现有些task分片小于合并的“目标大小”，这个时候，物理策略CoalesceShuffle Partitions就开始生效。这部分细节后面的aqe还会展开，这一讲因为focus在配置项，所以没说的特别细。

然后，对于你说的问题，我们来分类讨论。

第一种，假设是单表shuffle，比如reduceByKey这种，那么合并之后不影响，即便一个分片有多个key，也不要紧，只要保证同一个key的payload都在一个分区就行。

第二种，两表join。这个时候就有意思了，你说的担心就会成立，比如，外表大、内表小，外表合并之后，两个分区，第一个分区包含key=1、2、3，第二个分区，包含key=4、5、6；因为内表小，因此内表合并之后，一个分区就包含了1-6。其实，这样不要紧，为啥呢？如果你熟悉join实现原理的话（26讲会展开），在一个进程内，不论是nlj、smj还是hash join，外表，也就是驱动表，会驱动内表的“全量扫描”，带引号是因为效率不一样，smj、hj不用全量，但意思是一样的，就是不管内表有多少分区，都会被外表驱动着去被遍历。因此，不轮内表数据分配到了哪个分区，其实都还是在一个executor进程内，所以，不影响join逻辑，也不影响效率。nlj还是慢，hash join还是快，不会因为数据“看上去”打散了，而影响功能和性能。

手机徒手打字，可能没说特别清楚，有问题再留言哈～

1 6

kingcall
2021-04-09

其实我们的调优很多都是发生在数据规模比较大的情况下,对于比较大的shuffle 可以对下面的参数进行调节，提高整个shuffle 的健壮性

spark.shuffle.compress 是否对shuffle 的中间结果进行压缩，如果压缩的话使用`spark.io.compression.codec` 的配置进行压缩...

展开

作者回复: 没错，在超大规模的shuffle里，这几个参数都可以贡献任务执行的稳定性，不过compress我印象中默认是enabled。其他两个参数对于稳定性很有帮助~

3

辰
2021-04-13

这个aqe规则是在3.0版本才有的，但是我公司目前用的版本是2.2，有什么其他的参数设置吗

展开

作者回复: 爱莫能助，得升级到3.0~

2

斯盖丸
2021-04-06

spark.shuffle.file.buffer 和 spark.reducer.maxSizeInFlight 这两个配置项听起来应该越大越好，这样可以降低落盘的频次和数据shuffle的频次。请问老师这两个参数如果调太大会有什么副作用呢?

展开

作者回复: 内存资源宝贵且有限，这里多用一点，那里就少用一点。所以这里的缓存大了，其他地方比如执行、rdd cache、user memory就少了。

另外，buffer也并不是越大越好。它的主要作用是减少落盘和网络请求（IOPS）次数，但实际上，整体的数据量还是那么大，需要写入、拉取的数据，还是那么多，因此，不能指望把这两个buffer调大来显著提升执行效率。

1 2

耳东
2021-04-16

问下老师，如果某个表的大小是10m，但是它的条数超过了1千万条，这种表属不属于小表

展开

作者回复: 大小是相对的哈，参与Join的两张表，如果左右表的尺寸相差3倍以上，就可以考虑用“大表Join小表”的思路来解决这类关联问题了。我们在后面的第27讲，专门讲解“大表Join小表”的各种优化思路。

你说的情况，是和广播阈值作对比，就是能放进广播变量的，都可以算是小表。

所以，回答你的问题，10M我觉得算是小表了，和数据条目的数量没有关系，和整个数据集的内存存储大小有关系。Spark不会限制说，你的数据集条目超过了多少行，就不能做优化，主要还是看存储大小。

1

aof
2021-05-04

1. AQE的自动分区合并算法目前的实现，可能会造成合并之后，各分区数据不均衡，从而导致后续计算出现小的数据倾斜？合并算法是不是可以在得出合并前各分区大小之后，进行均衡的组合合并?
2. 因为AQE数据倾斜处理机制，是取中位数，那比如有10个分区：80MB 100 100 100 100 512 512 512 512MB，skewedPartitionFactor为5，skewedPartitionThresholdIn...

展开

天翼
2021-05-03

老师，请问一下，spark.sql.adaptive.nonEmptyPartitionRatioForBroadcastJoin 这个配置在官网的 Configuration 中没有找到，是改名了吗?

展开

作者回复: 好问题，看的很细~ 你是第一个深究这个参数的同学，赞一个~

这个是我的锅，我没有交代清楚，spark.sql.adaptive.nonEmptyPartitionRatioForBroadcastJoin，这是个internal的配置项，Spark社区并没有把它暴露给开发者，不过我们还是可以通过调整这个参数的设置，来调整Spark SQL的优化行为。

小灵芝
2021-04-29

老师您好呀！ 请教2个问题：

“因此，在不需要聚合，也不需要排序的计算场景中，我们就可以通过设置 spark.shuffle.sort.bypassMergeThreshold 的参数，来改变 Reduce 端的并行度”

…

展开

作者回复: 分别来说哈~

1. 比如像collect_list这种，也是“不需要聚合，也不需要排序的计算场景”，再比如coalesce(1, shuffle=True)，都属于这种。这两个是我立即就能想到的，我记得我之前好像汇总过类似的操作，等我抽时间找找，给你一份完整的操作列表。
2. groupByKey不需要聚合哈，这个是单纯的分组操作，没有聚合。

巧克力黑

2021-04-14

文章结尾图片中spark.shuffle.sort.bypassMergeThreshold为什么归类为Reduce端的参数呢?

展开

作者回复: 好问题，因为这个参数spark.shuffle.sort.bypassMergeThreshold，它指定的是Reduce阶段的并行度限制，所以归类到Reduce端参数。

但它实际控制的，是在Map阶段，能否byPass排序的操作。两者的关系确实有点蛋疼，但是理解就好哈~

zxk

2021-04-13

问题1：在用户不指定 RDD 大小推荐值或者分区数时，默认情况下我会将合并的推荐大小设置为总的数据量除以所有 executor 的核心数，然后在 map 端对将要 shuffle 到相同 partition 的分片先进行一次合并，减少需要 shuffle 的文件数量。
问题2：如果是那些需要某个分片全量数据才能得到正确结果的业务场景，那么自动数据倾斜可能会导致结果不符合预期。比如需要求该分片的某列数据的中位数，那么将该分片…

展开

作者回复: Map端的分区合并是个不错的思路~ 不过，这里有很多细节需要考虑，比如，你需要在Map阶段，就提前算出Reduce阶段的分区大小，才能决定说到底要Merge哪些分区；再者，

你需要修改BlockManager中维护关于Block的元数据，这样Reduce端在拉取数据的时候，才不会出错，诸如此类。

第二个问题答得很好，不过这个是“加盐”类操作的通病，就是搞不定需要全局排序的聚合操作，比如各种分位数的计算，当然，包括你说的中位数。我们来换个问法：和“手工加盐”相比，Spark的自动倾斜处理，有哪些性能隐患？不妨再思考一下哈~

1

对方正在输入。。。
2021-04-12

我来设计分区合并思路的话，会先按照每个partitionid的总体size来从大到小排序，然后按照背包算法来装包

展开

作者回复: 需要在分布式的上下文去考虑这个问题哈~

比如：
1. 排了序之后，按照什么规则合并分区？
2. 如果需要合并的partitions是跨Executors进程、跨节点的，怎么实现合并？Shuffle吗？

另外，可以在不引入额外Shuffle的情况下，优化分区合并的策略吗？等等，诸如此类~

maben996
2021-04-08

老师请教个问题，对于 spark.shuffle.sort.bypassMergeThreshold 这个参数，默认值为200。假如我的场景是需要排序的，但是reduce端的分区数又小于200，那这个参数会不会生效，触发bypass，绕过排序shuffle?

展开

作者回复: 不会的，你本身就需要排序，spark自然会利用排序机制来提升效率。

坏耗子
2021-04-06

问下老师，目前spark版本为2.3， 主要用spark sql为主进行开发，有什么好的sql方面的

调优思路?

展开

作者回复: 主要是充分利用catalyst和tungsten，另外再针对典型场景如joins，这些在后面的spark sql那部分都会讲哈～

onee

2021-04-06

请问spark.sql.shuffle.partitions是只有sparksql时有用还是所有？如果一个数据partition by成1024份了，那spark.default.parallelism和spark.sql.shuffle.partitions是不是都无效了?

展开

作者回复: 对，如果你明确repartition、或是HadoopRDD这种本身就是带分区的，spark会尊重你的意愿，那两个参数就没用了。 spark.sql.shuffle.partitions，是你在saprk sql开发框架之下，shuffle reduce阶段，默认的并行度。

白音

2021-04-06

问下老师，spark 2.x 的版本中 AQE 的这些特性都无法使用是吧?

展开

作者回复: 对，要升级到3.0